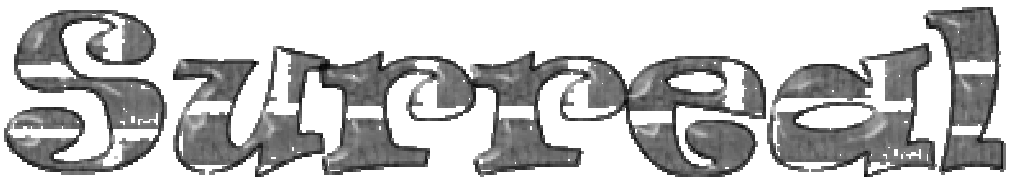
Inkorpotared Gamers Ko.

Technical Design Document



Written by Amanda Chaffin and Daniel Sabo

Version # 1.00

Monday, Febuary 9, 2009

Table of Contents

I. Design History

Version	Release Date	Comments
1.0	4.30.2009	Software Prototype

II. Technical Analysis

1. New Technology

None that we have determined.

2. Major Software Development Tasks

World Code.
AI Code.
Gravitational Orb Code.
Weapons.

3. Risks

None that we are not willing to assume.

4. Alternatives

There shall be several.

5. Estimated Resources Required

Time.

6. Technological Standards

Document Standards: HTML (with .html extensions that are not generated by MS Word—use Dreamweaver). PDF (must be readable by Adobe Acrobat and Apple Preview)

Coding Standard: http://serenity.uncc.edu/GameStudio/wp-content/uploads/2007/02/uncc-gdd-codingstandard-csharp.html

Presentations: MS PowerPoint or Apple Keynote, but they must also be provided in PDF

2D Graphics: PNG/JPEG

Game Files: Original design files for data should be in XML and include an XSD file (XML Schema) for validation. Other defined standards may be used as required by middleware and other component use

Diagrams: Use Visio or OmniGraffle and convert to PNG (include original source as well)

Compression: ZIP/RAR

III. Development Platform and Tools

1. Software

- Visual Studio C# 2008 Express Edition
- Microsoft DirectX SDK
    a. DirectX Graphics
    b. DirectShow
- Microsoft XNA Game Studio 3.0
- Gimp
- Paint Shop Pro 7
- Adobe CS3
- Xact Audio Authoring Tool
- Mappy Tile Map Editor

2. Hardware

- 1GHz Intel Pentium[i] III or AMD Athlon[ii] DirectX9[iii] compatible 32MB 3D graphics card with hardware transform and lighting
- 256MB RAM
- Microsoft Windows[iv] 2000/XP
- DirectX9 compatible sound card
- Mouse, keyboard, monitor

3. XBox Requirements

a. Xbox 360 and controller
b. XNA Creator's Club account
c. Internet access to the Xbox from the PC

4. Version Control Software

- Inkorporated Gamers Ko. is using Tortoise CVS Version 1.4.4. The CVS module is hosted on a UNCC Linux server accessed over the internet. All files are checked out from the server before changes are made, and only files that build a working executable should be committed to the server. All programmers should update their folders from the server daily.

IV. Delivery

Once the game is finished, there are two deliverables to hand in, one for the Xbox and one for the PC.

1. Medium

XBox – the game shall be deployed to the XBox via the internet and Microsoft Visual Studios C# Express Edition's Deployment Software which comes with the Visual Studio XNA package.

PC – the game shall be compiled with the NullSoft Scriptable Install System and burned to a CD/DVD for the class turn in. The NullSoft Scriptable Install System shall contain all the information needed to install and run the game on any compatible PC.

2. Required Hardware and Software

1. XBox360
    i. Controller
    ii. PC with the game connected to the XBox
    iii. Creators' Club Account with XNA
2. PC
    i. CD/DVD Rom
    ii. Mouse
    iii. Keyboard
    iv. Compatible Development Machine

3. PC Installation Instructions

1. Insert the Install disk into the CR-Rom drive and copy the contents down to a local directory.
2. Run the surreal.exe to install the game. This shall download and install the DirectX SDK as well as the XNA platform to the machine.
3. Play the game!

4. XBox Deployment Configuration

1. Load an Xbox 360 Game project into XNA Game Studio.
2. Make sure the Output window is visible so you can follow the progress of your deployment in detail, and so you can tell where a problem occurred, if one does. To display the Output window, from the View menu, click Output, or press CTRL+W and then press the letter O key.
3. From the Games page of the Xbox 360 Dashboard, select Games Library and press the A controller button.
4. From the Games Library page, select My Games and press the A controller button.
5. Select XNA Game Studio Connect and press the A controller button.

6. Select Launch and press the A controller button. This displays the Waiting for computer connection page.
7. In XNA Game Studio, deploy the game by right-clicking the solution name and selecting Deploy Solution or selecting Deploy Solution from the Build menu.During this process, both the Output window of XNA Game Studio and the Connect to Computer screen display the deployment progress by displaying a list of the deployed files and additional information.
8. Press B to return to the Games Library. From there, select My Games and press the A controller button. This displays a list of available games.
9. Select the game you just deployed from this menu and press the A controller button, then choose Play Game and press the A controller button again to start the game.

V. Game World

1. Technical Specification

Surreality is the world that the character moves and interacts with. It handles the creation and updating of the objects, characters and AI as well as their destruction. The actual construction and implementation of the world map and art are handled by this as well.

2. Design

(1) (R-GW-1) A 2d side scroller world
(2) (R-GW-2) The world shall be based on a tile engine

3. Objects

1. Features
    1. Vector2 Position
    2. Float Velocity
    3. Vector2 Origin
    4. Int Object size
    5. Int Object ID
    6. Float Radius
    7. Float Mass
    8. Int Tile number

2. Details
    1. Bounding box create the boundaries around the object for collision checking
    2. Collision checks to see if objects have collided
    3. Movement takes physics measurements such as velocity and calculates movement
    4. Object initialization methods sets all necessary starting data
    5. Object Update methods handles the renewal of data and its changes
    6. Object Draw methods draws all the necessary art work and graphics to the screen
    7. Animation method for animating the character spritesheets

4. Game Objects

1. Features
    1. Gravitational Orbs
    2. Human
    3. AI
    4. Bat
    5. Resources
    6. Inherits Objects*

2. Details

i. Gravitational Orbs

1. Features

a. An orb graphic
b. A gravitational force which effects only the player

2. Details

a. A graphic image ranging from 64x64 to 256x256
b. (R-GW-7a)A gravitation equation that draws the player onto the surface
c. (R-GW-7b)Upon the appropriate button press "jump button" the gravity shall release allowing the user to escape the orbs gravitation field

ii. Human

Refer to Section IX. Game Characters

iii. AI

Refer to Section IX. Game Characters

iv. Bat

Refer to Section IX. Game Characters

v. Resources

1. Features
   a. Interactive objects
   b. Health
2. Details
   a. Interactive Objects
      1. (R-GW-8a)There are four graphics that are 32x32
         1. Square Graphic
         2. Circle Graphic
         3. Triangle Graphic
         4. Water Drop Graphic
      2. (R-GW-8b)Two states
         1. Active handles
            a. Normal Collision
            b. Can be picked up with gravity gun
            c. Can be shot with mortar
            d. Can be pushed by player
            e. Effected by gravity
            f. Can kill enemies
         2. Inactive handles
            a. Not effected by gravity
            b. Collides with the mortar
            c. Collides with gravity gun
            d. Does not collide with other objects
            e. Does not collide with Character
            f. Can be picked up by gravity gun
            g. Turns to active state by collision with mortar
            h. Turns to active state by collision with gravity gun

3. (R-GW-8c)Water drop object
    a. Has a shader effect for water
4. Tomato
    a. (R-GW-9)A tomato graphic 32x32
    b. Increases players health
    c. (R-GW-10)If within close enough proximity flips the bat side kick to tomato state

5. Travel

1. Features
    1. Horizontal movement
    2. Vertical movement
    3. Bat grabbed movement
    4. Bat grabbed puzzle
2. Details
    1. (R- GW-18)Horizontal movement
        a. One half tile along the x axis per button press
    2. (R-GW-20)Vertical movement
        a. Two tiles along the y axis per button press
    3. (R-GW-19a)Bat grabbed movement
        a. Move in random direction and tile distance as dictated by the random bat flying algorithm for 2 seconds
    4. (R-GW-19b)Bat grabbed puzzle
        a. Move to set location according to point

6. Scale

1. (R-GW-21)Screen 1024 x 768
2. (R-GW-22)Tiles 32 x 32

7. Time

1. Features
    1. Freeplay Mode
    2. Timed Mode

2. Details
    1. In clock mode, there is a set time (5 minutes) to beat the level.
        1. The clock shall be created with System.Diagnostics.StopWatch as a global variable
        2. The game1 class shall start the clock.
        3. The game1 class shall stop the clock.

    2. Free play mode the clock counts but no longer is a requirement to beat the level.

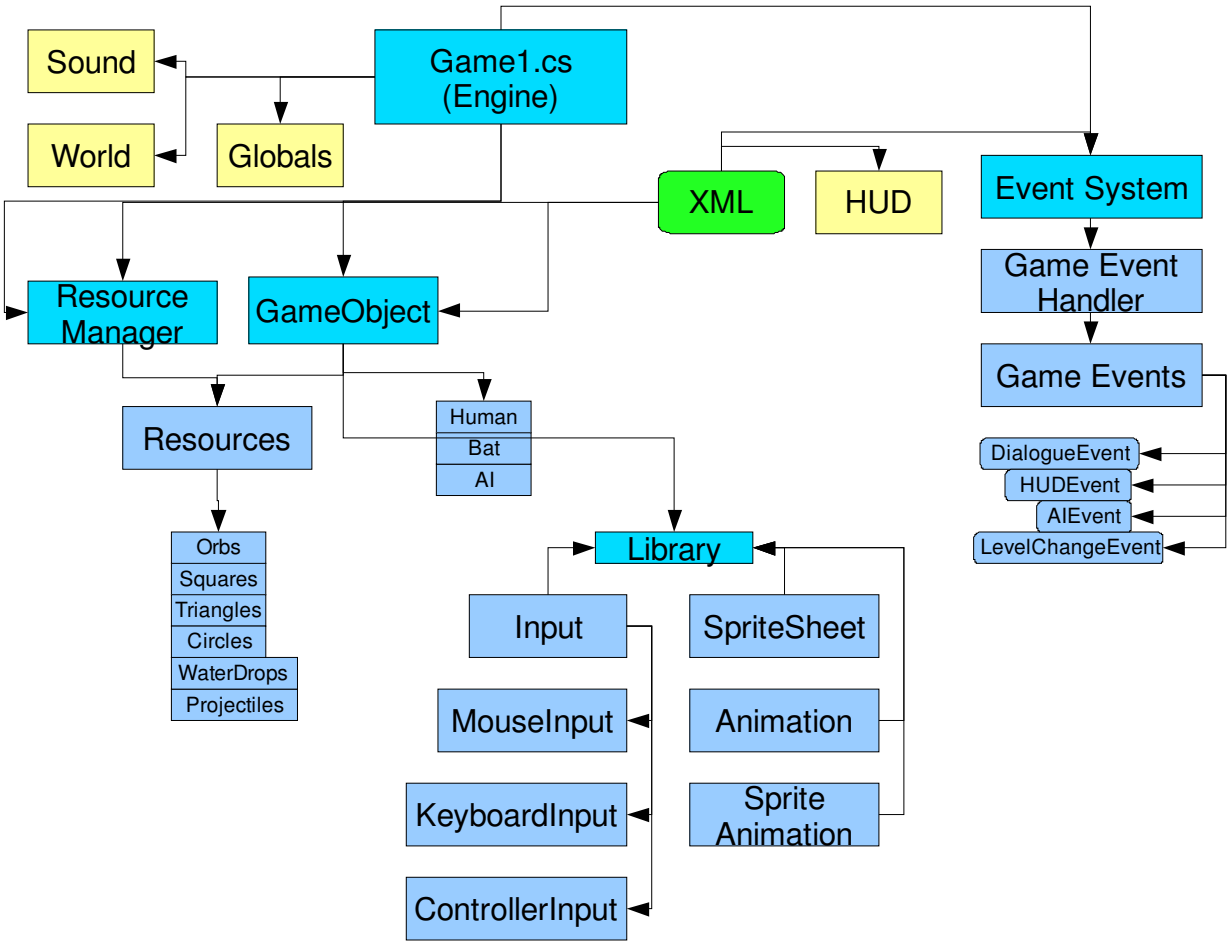
V. Game World

1. Technical Specification

The game engine controls the main game loop for the game, the workings of the game world to include on screen (game objects, the player character, etc) and off screen programming (physics, collision, AI, etc).

2. Design

The game engine includes the tile engine (R-GE-1-5), world editing (R-GE-6-9), the event system (R-GE-10-15), the resource manager(R-GE-16-18), sound system (R-GE-17-19), water R-GE-20-22), and collision detection (R-GE-23).



3. Tile Engine

1. (R-GE-1) – The engine shall be based on an array data structure.
    1. (R-GE-1a) – The array shall be 0 based.
2. (R-GE-2) – There shall be 7 levels in the game.
    1. (R-GE-2a) – Each level shall be 1024 by 768 pixels.
3. (R-GE-3) – Tile set one shall be the background image.
4. (R-GE-3) – Tile set two shall be the bricks for the player to walk on.

Please see Appendix World for Code

4. World Editing

1. (R-GE-6) – The World shall be hand drawn with a Bamboo Wacom Tablet.
1. (R-GE-6a) – The World shall be saved using Gimp.
2. (R-GE-6a) – The World shall be saved as a .png
2. (R-GE-7) – Mappy shall be used to generate the maps.
3. (R-GE-8) – Mappy shall be used to create the tile maps.
4. (R-GE-9) – Mappy shall be used to create the underlying array structure of the world.

5. Event System

1. Features
    1. Loading of events shall be handled by XML – refer to Section XV for details.
    2. (R-GE-10) – There shall be four event types.
    3. (R-GE-11) – Game Events shall fire when trigger conditions are met.
    4. (R-GE-12) – The event system shall maintain a static board evaluator known as the current game conditions.
    5. (R-GE-13) – The current game conditions shall be changed by any part of the engine that can see the Event System.
        a. (R-GE-13a) – Game object collision with game object.
        b. (R-GE-13b) – A preset time in the game.
        c. (R-GE-13c) – Upon other events being fired.
    6. (R-GE-14) – The Event System shall remove all items from the list before swapping levels.
    7. (R-GE-15) – The Event System shall load a new level upon the old one being closed.

2. Details
    1. AIEvent
        a. (R-GE-10a) The Event shall move the AI to a preset location.
        b. (R-GE-1b) The Event shall change the AI's state in the state machine.
    2. HUDEvent
        a. (R-GE-10c) – The Event shall add an image/text to the HUD.
        b. (R-GE-10d) – The Event shall remove images or text on the HUD.
        c. (R-GE-10e) - The Event shall replace images or text on the HUD .
    3. DialogueEvent
        a. (R-GE-10f) – The event shall add a preset dialogue with a speaker and a message to the HUD that shall run for a preset amount of time.
    4. LevelChangeEvent

a. (R-GE-10g) - The Event shall call the level change for the game.

3. GameEvent Scaffolding Code
   Please see Appendix B for Code
4. GameEventHandler Scaffolding Code
   Please see Appendix C for Code
5.. DialogueEvent Scaffolding Code
   Please see Appendix D for Code

6. Resource Manager

1. Features
   a. Loading of resources shall be handled by XML – refer to Section XV for details.
   b. (R-GE-16) – The resource manager shall handle the resources of the game.
   c. (R-GE-17) – AI or Human/Bat shall be able to consume/use all resources.
   d. (R-GE-18) – Additional resources shall be added by the event system.

2. Details
   a. (R-GE-16a) – Basic Game Objects shall be stackable and shootable.
      i. Square.
      ii. Triangle.
      iii. Circle.
   b. (R-GE-16b) – Water Droplets shall fill the pit. Refer to Section 8 in Game Engine.
   c. (R-GE-16c) – Tomatoes shall add health to the player and attract "Teh Bat." Refer to Section IX for details.

3. Resource Manager Scaffolding Code
   Please see Appendix E for Code
4. Resources Scaffolding Code
   Please see Appendix F for Code

7. Sound System

1. Features
   1. (R-GE-17) – A sound manager shall be used to control all game audio.
   2. (R-GE-18) – A song handler shall be used to control the main theme.
   3. (R-GE-19) – A cue handler shall be used to control all sound effects.
   4. Sound System Scaffolding Code
      Please see Appendix G for Code
2. Details
   1. (R-MS-1) – The game shall have background music.
   2. (R-MS-2) – The game shall have sound effects.
      a. (R-MS-2a) – Player action.
      b. (R-MS-2b) – Enemy action.
      c. (R-MS-2c) – "Teh Bat" action.
      d. (R-MS-2d) – Randomly.
   3. (R-MS-3) – Heartbeat.
   4. (R-MS-4) – Bat shall have noises.

5. (R-MS-5) – Player shall have noises.
6. (R-MS-6) – Object shall have noises.
7. (R-MS-7) – Enemy shall have noises.

8. Water

1. (R-GE-20) – Water droplets shall collide with projectiles from the human.
2. (R-GE-21) – Water droplets shall fall into the pit of spikes.
3. (R-GE-22) – Water droplets shall, when the player has shot 4, shall turn into a collision plane that the player can traverse.

9. Collision Detection

1. (R-GE-23) – Collision detection shall be constructed in the Game Object parent class.
2. (R-GE-23a) – Water droplets shall fall into the pit of spikes.
3. (R-GE-23b) – Water droplets shall, when the player has shot 4, shall turn into a collision plane that the player can traverse.

VII. Rendering System

1. Technical Specifications

The rendering system shall harness the power of the XNA graphics device to handle the rendering so that the developers do not have to code the specifics. The game shall be rendered in 2D with a simplistic camera system. All sprites in the game shall use the SpriteSheets class to load the images from the spritesheets. All sprites that need to be animated (players, AI, game objects) will use the animation system.

2. 2D/3D Rendering

1. (R-RS-2) - Rendering shall use the built-in graphics rendering pipeline provided by the XNA game development platform.

2. (R-RS-3) - All drawable items in the game shall inherit from the DrawableGameComponent which allows the developers to use the same draw functionality across the entirety of the code base.

3. Camera

1. (R-RS-4) – The camera shall have minimal functionality.
2. (R-RS-5) - The camera shall be fixed to the player character.
3. (R-RS-6) – The camera shall be centered over the character at all times.

4. Sprite Sheets

1. (R-RS-7) – All sprites will use the spriteSheet class.
2. (R-RS-8) – The spritesheet contains all information to draw the sprite
    a. (R-RS-8a) – Image location.
    b. (R-RS-8b) – Sprite height
    c. (R-RS-8c) – Sprite width.
    d. (R-RS-8d) – Animation count.
3. (R-RS-9) – The spritesheet shall be called from constructor of the calling class.
4. (R-RS-10) – The spritesheet shall call Load() from the CallingClass.LoadContent()
5. (R-RS-11) – The spritesheet shall use getTextureRectangle(index) to fetch the rectangle.
6. (R-RS-12) – The rectangle index is 0 based.

5. Sprite Sheets Scaffolding Code

Please see Appendix H for Code

6. Animation System

1. (R-RS-9) – Spriteanimation shall run each animation for each sprite.
2. (R-RS-10) – The animation class shall be build each animation for each sprite.
    a. (R-RS-10a) – The animation class shall store each animation inside the animation dictionary data structure for ease of access.
    b. (R-RS-10b) – The data for the animations shall be stored in flat text files.
        i. Number of frames.

ii. Sprite location.
iii. Bounding box.
iv. Offset.

c. (R-GE-10c) – After receiving the call from the animating object, the animation shall run the animation on the object.
d. (R-GE-10d) – The animation object shall house the animations.
e. (R-GE-10e) – The animation class shall load animations.
f. (R-GE-10f) - The animation class shall iterate animations.
g. (R-GE-10g) - The animation class shall close animations.

7. Animation Scaffolding Code

Please see Appendix I for Code

VIII. World Layout

1. World Map

1. Features
    1. (R-WL-1a)A tile based map
    2. (R-WL-1b)An image background
    3. Set locations for event tripping
    4. (R-WL-2)Puzzle Locations
    5. (R-WL-3)Refer to Appendix II

1. Details
    1. For creation Refer to section 4 in Game Engine
    2. There are 10 locations set throughout the world which set off events
        1. Refer to Appendix for Details
    3. (R-W-4) - There are 8 world puzzles.

2. World Puzzles

1. (R-WL-5) The 'Tetris' puzzle - the player has to build a staircase using the gravitational gun, the mortar, or his body to move the blocks so that he can jump up to the next level.

2. (R-WL-6) 'Teh Bat' Cave – the player must locate, collect, and use the gravitational gun on 'Teh Bat' in order to fly up to the next level.

3. (R-WL-7) The 'Orb' puzzle – the player has to jump from orb to orb, allowing the gravity of the orb to catch him in mid-flight and the rotation of the orb to throw him in the right direction.

4. (R-WL-8) 'Teh Bat Oops' puzzle – while trying to be helpful, 'Teh Bat' knocks down one of the orbs the player needs to jump on, causing the player to fall down to the next level.

5. (R-WL-9) The Water puzzle – the player must shoot down enough water droplets to fill a hole in the ground that he can traverse it.

6. (R-WL-10) Second 'Orb' puzzle.

7. (R-WL-11) The 'Other Bat' Cave puzzle – annoyed at Dude, 'Teh Bat' flies off into a cave and Dude must rescue 'Teh Bat' from the clutches of the other bats inside the cave.

8. (R-WL-12) The Final Battle – the player must defeat the Flying Spaghetti Monster in a battle to the death.

IX. Game Characters

1. Player Characters

1. Features
    1. (R-GC -1) Player character “Dude”
    2. (R-GC -2) Side kick character “Teh Bat”
2. Details
    1. Player
        a. (R-GC -1a) A player image 32x32
        b. (R-GC -1b) The ability to jump
        c. (R-GC -1c) The ability to use the mortar
        d. (R-GC -1d) The ability to use the gravity gun
        e. (R-GC -1e) A series sprite animations
        f. (R-GC -1f) Health
        g. (R-GC -1g) Player input
        h. (R-GC -1h) Sound effects
        i. (R-GC -1i) Has the ability to walk
        j. (R-GC -1j) Bat flight
    2. Side kick Character
        a. (R-GC -2a) A series of sprite animations
        b. (R-GC -2b) 4 different state
            i. Idle
                1. Does nothing
            ii. Tomato
                1. Flies to tomatoes on world view and hovers within a 3 tile radius
            iii. onHuman
                1. Flies to the player and hovers within a 3 tile radius
            iv. Flight
                1. Moves to a fixed point in the world when grabbed by the gravity gun at certain locations
        c. (R-GC -2c) Sound effects

2. AI

1. Features
    a. A crawler AI
    b. A bouncer AI
    c. A Vampire Bat AI “enemy bat”
    d. FSM Boss AI
2. Details
    a. (R-GC-3) A crawler AI
        a. (R-GC-3a).2 states
            i. Idle
                1. Random path generator
            ii. Spotted player
                1. Straight line to player algorithm
        b. (R-GC-3b).When collides with player causes damage

c. (R-GC-3c).Colliding with mortar destroy unit
d. (R-GC-3d).Can be picked up by gravity gun
e. (R-GC-3e).Land based movement
f. (R-GC-3f).Light damage
g. (R-GC-3g).Fast speed

e. (R-GC-4)A bouncer AI
   a. (R-GC-4a)2 states
      i. Idle
         1. Random path generator
      ii. Spotted player
         1. Straight line to player algorithm
   b. (R-GC-4b)When collides with player causes damage
   c. (R-GC-4c)Colliding with mortar destroy unit
   d. (R-GC-4d)Can be picked up by gravity gun
   e. (R-GC-4e)Land bouncing based movement
   f. (R-GC-4f)Heavy damage
   g. (R-GC-4g)Medium speed
f. (R-GC-5)A Vampire Bat AI “enemy bat”
   a. (R-GC-5a)2 states
      i. Idle
         1. Random path generator
      ii. Spotted player
         1. Straight line to player algorithm
   b. (R-GC-5b)When collides with player causes damage
   c. (R-GC-5c)Colliding with mortar destroy unit
   d. (R-GC-5d)Can be picked up by gravity gun
   e. (R-GC-5e)Flying unit based movement
   f. (R-GC-5f)Medium damage
   g. (R-GC-5g)Fast speed
g. (R-GC-6) FSM Boss AI
   a. (R-GC-6a)2 states
      i. Idle
      ii. Spotted player
         1. Begin attack sequence
   b. (R-GC-6b)When collides with player causes damage
   c. (R-GC-6c)Colliding with mortar hurts
   d. (R-GC-6d)Shoots projectiles at player

3. Human and AI Scaffolding Code

Please see Appendix J for Code

X. HUD Technical Specifications

1. Features
    1. Loading of resources shall be handled by XML – refer to Section XV for details.
    2. (R-UI-1) – The HUD displays on screen at all times.
    3. (R-UI-2) – The HUD is loaded to a preset resolution (1280x1024).
    4. (R-UI-3) – Upon load, the HUD shall be resized to match the game viewport.
    5. (R-UI-4) – The HUD shall display a character icon.
    6. (R-UI-5) – The HUD shall display the player health icon.
    7. (R-UI-6) – The HUD shall display a countdown of time from 5 minutes to zero.

2. Details
    1. (R-UI-4a) – The character icon shall be animated
    2. (R-UI-4b) – The Character icon shall smile.
    3. (R-UI-4c) – The Character icon shall scowl.
    4. (R-UI-4d) – The Character icon shall have a neutral expression.
    5. (R-UI-5a) – Health is 100 points.
    6. (R-UI-5b) – Health decreases when player takes damage.
    7. (R-UI-5c) – Health increases when player finds tomato.

1. HUD Scaffolding Code

Please see Appendix HUD for Code

2. Screen Object Scaffolding Code

Please see Appendix ScreenObject for Code

XI. Weapons

1. Gravity Gun

1. (R-W-1) - Features
    1. An image for the gun
    2. An ray image
    3. A shooting method
    4. (R-W-1a,b)A grabbing method
    5. (R-W-1c)A bat grabbing method
2. Details
    1. The gun shall rotate around the player according to targeted position
    2. The shooting method shall handle the projection of the gravity gun
        a. Shoot a constant beam
        b. Beam length 3 tiles long
    3. (R-W-1d)A grabbing method shall handle the selecting of Interactive Objects
        a. Highlight the object using a shader
        b. Cancel gravity
        c. (R-W-1e)Give full control over object
        d. (R-W-1e)Give free control on a 3 tile radius
        e. (R-W-1f)Upon release give control of object back to Game Object
    4. (R-W-1g)A bat grabbing method which shall grab the bat
        a. Shall allow characters to fly according to bats random flying patterns
        b. In certain locations a predetermined flight pattern and destination is given

2. Mortar

1. (R-W-1) - Features -
    1. An image for the gun
    2. A projectile image
    3. (R-W-2a)A shooting method

2. Details
    1. A shooting method
        a. Hold button down longer increase trajectory
            i. Min distance 1 tile
            ii. Max distance 10 tiles
        b. Creating the projectile
        c. Launching the projectile
        d. (R-W-2b) Switches Interactive Object state to Active
        e. (R-W-2c)On projectile Collision cause enemy damage

XIII. Single Player Game

1. Game Play

1. Features
    1. Goal Oriented
    2. Use of actions to complete the current goal
    3. Walk
    4. Jump
    5. Move Objects
    6. Shoot objects with a “gravity gun”
    7. Mortar
    8. Flying with the aid of “Teh Bat”
2. Details
    1. Fictitious World
    2. Run by “Vampire Bat” Clan
    3. In conflict with “Flying Spaghetti Monster” for natural resources
    4. Main Player: a third party that is designated to restore balance

2. Victory Conditions

1. (R-SPG-1) Five minute time limit.
2. (R-SPG-2) Defeat the main boss.
3. (R-SPG-3) Accomplish the puzzles and destruction of the boss within five minutes.
4. (R-SPG-4) Defeat Puzzles

XIV. Control Technical Specifications

1. Features
    a. Keyboard input
    b. Xbox 360 controller input
2. Details
    a. Keyboard
        i. WASD directional movement
        ii. “space bar” jump
        iii. Left click shoot projectiles
        iv. Right click shoot gravity gun
        v. Esc pause
        vi. Mouse movement aiming
    b. Xbox 360
        i. Left analogue stick directional movement
        ii. A button to jump
        iii. Right trigger to shoot projectiles
        iv. Left trigger to shoot gravity gun
        v. Start to pause
        vi. Right analogue stick for directional aiming

XV. XML Handling

1. Technical Specifications

There shall be three systems in the engine that use XML to load content into the game (Events [R-XML-2], Resources [R-XML-3], and HUD[R-XML-4]).

1. R-XML-1 - XML content shall be loaded via a preformatted structure.
2. R-XML-1a – The preformatted structure…Please see Appendix M for Code
3. R-XML-1b - XML loading shall always be wrapped in a try catch statement.

2. Details

1. (R-XML-2) – All events shall be loaded via XML.

- (R-XML-2a) – The DialogueEvent shall have the following XML Schema…

```
<GameEvent typeID ="DialogueEvent" speaker ="string"
message="string"
texturePath ="ArtAssets/path" time ="int"
node ="int" coins ="int" health ="int"/>
```

- (R-XML-2b) – The AIEvent shall have the following XML Schema…

```
<GameEvent typeID ="AIEvent" ID="int" drawable ="bool"
destX
="int" destY="int" destZ ="int" node ="int" coins ="int"
health ="int"/>
```

- (R-XML-2c) – The HUDEvent shall have the following XML Schema…

```
<GameEvent typeID ="HUDEvent" name ="string" texture =
"ArtAssets/path" texX ="int" texY="int" draw ="bool"
node ="int" coins ="int" health ="int"/>
```

- (R-XML-2d) – The LevelChangeEvent shall have the following XML Schema…

```
<GameEvent typeID="LevelChangeEvent" node="int" coins =
"int" health ="int" nextLevel="int"/>
```

2. (R-XML-3) – All preloaded game objects shall be loaded via XML.

- (R-XML-3a) – Basic resources XML Schema

```
<GameObject typeID ="string" name ="string" texture =
"ArtAssets/path" texX ="int" texY="int" draw ="bool"
gravity ="bool" mass ="float"/>
```

- (R-XML-3b) – Nonstandard resource XML Schema

```
<GameObject typeID ="string" name ="string" texture =
"ArtAssets/path" texX ="int" texY="int" draw ="bool"
gravity ="bool" mass ="float" rotation ="bool"
rotationspeed ="float"/>
```

3. (R-XML-4) – All HUD Images and Text shall be loaded via XML.

- (R-XML-4a) – HUD image XML Schema

```
<ImageDisplay nameID ="string" texturePath
="ArtAssets/path" posX ="#" posY ="#" colorR ="#" colorG
="#" colorB ="#" colorA ="#" toDraw ="bool" imageReference
="string" position =" string "/>
```

- (R-XML-4b) – HUD text XML Schema

```
<TextDisplay nameID ="string" text ="string" posX ="int"
posY=" int" colorR ="int" colorG ="int" colorB ="int"
colorA ="int" toDraw ="bool" imageReference ="string"
operation ="string" offsetX ="int" offsetY="int"/>
```

XVII. Appendices

World

```
namespace GameOrDie.EventSystem
{
    /// <summary>
    /// This is a game component that implements IUpdateable.
    /// </summary>
    public class World : DrawableGameComponent
    {
        #region Fieldsf
        private int[,] mapArray;
        #endregion

        #region Initialization
        public World(Game game)
            : base(game)
        {
        }
        /// <summary>
        /// Query other services and base.Initialize enumerates through
the
        /// component and Initializes them as well.
        /// </summary>
        public override void Initialize()
        {
            resourceMgr.Initialize();
            base.Initialize();
        }
        public void LoadContent()
        {
            resourceMgr.LoadContent();
            base.LoadContent();
        }
        #endregion

        #region Update
        public override void Update(GameTime gameTime)
        {
            resourceMgr.Update(gameTime);
            UpdateInput();
            base.Update(gameTime);
        }

        private void UpdateInput()
        {
        }
        #endregion

        #region Draw
        public override void Draw(GameTime gameTime)
        {
            DrawResources(gameTime);
```

```
            base.Draw(gameTime);
        }
        private void DrawResources(GameTime gameTime)
        {
            resourceMgr.Draw(gameTime);
        }
        #endregion

        #region Helper Methods
        /// <summary>
        /// Populates mapArray
        /// </summary>
        private void LoadLevel()
        {
            #region Array data
            #endregion
        }
        public Vector2 ScreenToWorld(Vector2 screenPosn)
        {
            return worldPosn;
        }
        private Vector2 WorldToScreen(Vector2 worldPosn)

            return screenPosn;
        }
        #endregion
    }
}
```

GameEvent

```
namespace GameOrDie.EventSystem
{
    /// <summary>
    /// Handles each GameEvent's Conditions, the SBE
(currentGameConditions)
    /// Is the abstract of each GameEvent
    /// </summary>
    public class GameEvent
    {
        #region GameEvent State Variables
        #endregion

        #region Static Board Evaluator (SBE) State Machine
        #endregion

        /// <summary>
        /// CurrentGameConditions Constructor
        /// </summary>
        public GameEvent()
        {
        }

        /// <summary>
        /// Constructor for each Game Event
```

```
        /// </summary>
        /// <param name="node">XML Node</param>
        public GameEvent(XmlNode node)
        {
        }

        public virtual void FireEvent()
        {

        }
        public virtual void DrawEvent()
        {

        }

    }
}
```

GameEventHandler

```
 namespace GameOrDie.EventSystem
{
    /// <summary>
    /// Handles the GameEvents from the game
    /// </summary>
    public static class GameEventHandler
    {
        public static List<GameEvent> GameEvents;
        public static GameEvent CurrentGameConditions;

        /// <summary>
        /// Creates the GameEvents based on the data from the XML files
        /// </summary>
        public static void LoadEventsXml(string fileName)
        {
            GameEventHandler.killEventList();

            GameEvents = new List<GameEvent>();
            CurrentGameConditions = new GameEvent();
            /// Insert XML Loading here
        }

        /// <summary>
        /// Update the gameEvents - called from GameEvent.cs
        /// Handles running the visualization on the event
        /// and removal when dead
        /// </summary>
        public static void UpdateEvents()
        {
        }
        /// <summary>
        /// Checks to see if the event has been triggered
        /// </summary>
        /// <param name="theEvent">Instance of the Event</param>
```

```
        /// <returns>Is the Event Triggered?</returns>
        public static bool isEventTriggered(GameEvent theEvent)
        {
        }
        /// <summary>
        /// Level reset
        /// </summary>
        public static void killEventList()
        {
        }
    }
}
```

DialogueEvents

```
 namespace GameOrDie.EventSystem.EventTypes
{
    /// <summary>
    /// Instance of a DialogueEvent
    /// Extends GameEvent
    /// Used to update the Dialogue graphics and text on the HUD
    /// </summary>
    public class DialogueEvent : GameEvent
    {
        /// <summary>
        /// Create for the dialogue event
        /// </summary>
        public DialogueEvent(XmlNode node )
            :base(node)
        {
        }
        /// <summary>
        /// Runs the dialogue on the HUD for the specified time
        /// </summary>
        public override void FireEvent()
        {
        }
    }
}
```

ResourceManager

```
namespace GameOrDie.GameObjects
{
    /// <summary>
    /// This is a game component that implements IUpdateable.
    /// </summary>
    public class ResourceManager : DrawableGameComponent
    {
        #region Fields
        SpriteSheet spriteSheet;
        public List<Resource> resourceColl;
        #endregion
```

```
    #region Initialization
    public ResourceManager(Game game)
        : base(game)
    {
        // TODO: Construct any child components here
    }

    /// <summary>
    /// Allows the game component to perform any initialization it
needs
    /// to before starting to run.  This is where it can query
    /// for any required services and load content.
    /// </summary>
    public override void Initialize()
    {
        // TODO: Add your initialization code here
        base.Initialize();
    }

    public void LoadContent()
    {
        spriteSheet.Load();
    }
    #endregion

    #region Update

    /// <summary>
    /// Allows the game component to update itself.
    /// </summary>
    public override void Update(GameTime gameTime)
    {
        // TODO: Add your update code here
        base.Update(gameTime);

        foreach (Resource resource in resourceColl)
        {
            resource.Update(gameTime, player);
        }
    }

    #endregion
    public override void Draw(GameTime gameTime)
    {
        foreach (Resource resrc in resourceColl)
        {
            resrc.Draw(spriteSheet, gameTime);
        }
    }
    #region Properties

    public SpriteSheet spriteSheet
    {
        get { return spriteSheet; }
    }
```

```
        #endregion

    }
}
```

Resource

```
namespace GameOrDie.GameObjects.Items
{
    public abstract class Resource : GameObject
    {
        #region Fields
        protected bool consumed;
        protected bool isUsable;
        protected bool active;
        #endregion

        #region Properties

        public int SpriteIndex
        {
            get { return tileNumber; }
        }
        public bool IsUsable
        {
            get { return isUsable; }
            set { isUsable = value; }
        }
        public bool Consumed
        {
            get { return consumed; }
            set { consumed = value; }
        }

        /// <summary>
        /// Resource still exists or exhausted
        /// </summary>
        public bool Active
        {
            get { return active; }
            set { active = value; }
        }
        #endregion

        public Resource(Game game,Vector2 posn):base(game)
        {
            Consumed = false;
            worldPosition = posn;
        }

        /// <summary>
        /// Updates the player depending on the resource.
        /// </summary>
        /// <param name="gameTime">Snapshot of the time</param>
```

```
        /// <param name="player">Player whose attribs are changed as
per
        /// resources</param>
        public abstract void Update(GameTime gameTime, Human player);

        public abstract void Action(GameTime gameTime, Human player);
    }
}
```

SoundManager

```
namespace GameOrDie.Audio
{
    /// <summary>
    /// AudioManager is used for sound of course and was extracted from
    ///the GameScreen class. This class is a based off a Singleton
Pattern
    /// so that there is only 1 instance of it floating around.
    /// To call this class you must use GetInstance().whatever
    /// </summary>
    public class AudioManager : GameComponent
    {
        // Variable to get a sound flag
        static bool soundsOnFlag = true;

        // Audio API components.
        AudioEngine audioEngine;
        WaveBank waveBank;
        public static SoundBank soundBank;

        //Accessor for the sound flag
        public bool SoundsOnFlag
        {
            get
            {
                return soundsOnFlag;
            }
            set
            {
                soundsOnFlag = value;
            }
        }
        public AudioManager(Game game):base(game)
        {
            Game.Services.AddService(typeof(AudioManager), this);
            audioEngine = new AudioEngine("Content/Sound
              Assets/Name_Of_Audio_Engine.xgs");
            waveBank = new WaveBank(audioEngine, "Content/Sound
              Assets/Name_Of_Wave_Bank.xwb");
            soundBank = new SoundBank(audioEngine, "Content/Sound
              Assets/Name_Of_SoundBank.xsb");
        }
        public static void PlayClick()
        {
```

```
            if (soundBank != null && AudioManager.soundsOnFlag == true)
            {
                soundBank.PlayCue("Name_Of_Sound_Effect");
            }

        }
        public static void PlayTheme()
        {
            if (soundBank != null && AudioManager.soundsOnFlag == true)
            {
                soundBank.PlayCue("Name_Of_Theme_Song");
            }

        }

    }
}
```

SpriteSheet

```
namespace Library.SpriteBoard
{
    //SpriteSheet should not containt any alpha layer tile in 1st row

    //Remove Redundant Fields
    //Texture String a variable
    //Amend Load with texture string

    public class SpriteSheet
    {
        #region Fields
        #endregion

        #region Properties

        public Texture2D SpriteSheeet
        {
            get
            {
                return spriteSheet;
            }
        }

        public int TileWidth
        {
            get
            {
                return tileWidth;
            }
        }

        public int TileHeight
        {
            get
```

```
            {
                return tileHeight;
            }
        }

        #endregion

        #region Initilization

        public SpriteSheet()
        {

        }


        void Initialize()
        {

        }

        //Load texture
        public void Load()
        {
            spriteSheet = content.Load<Texture2D>(spriteName);
        }

        #endregion
    }
}
```

Animation

```
namespace GameOrDie.GameObjects
{
    public class Animation
    {
        #region Fields
        // Animation frames
        // X and Y offsets for frame to frame
        // How long the frame will last
        // Timer for tracking the frame
        // Current frame
        // Indicates if the current frame is in its first pass
        #endregion // Fields

        #region Properties

        /// <summary>
        /// Get the current frame's rectangle
        /// </summary>
        public Rectangle CurrentFrameRectangle
        {
        }
        /// <summary>
```

```
        /// Get the offset for the current frame
        /// </summary>
        public Vector2 CurrentOffset
        {
        }
        /// <summary>
        /// Accessor for the Framerate of the animation
        /// </summary>
        public int FPS
        {
        }
        #endregion // Properties

        #region Initialization
        /// <summary>
        /// Constructor - read in from text file
        /// </summary>
        public Animation(string dataFilename)
        {
        }

        # endregion // Initialization

        # region Update
        /// <summary>
        /// Update
        /// </summary>
        /// <param name="gameTime"></param>
        public bool Update(GameTime gameTime)
        {
            return result;
        }

        # endregion // Update

        # region Methods
        /// <summary>
        /// Reset frame and animation timers
        /// </summary>
        public void resetAll()
        {
        }
        #endregion // Methods

    }
}
```

Human

```
namespace GameOrDie.GameObjects.Player
{
    // RENAME FOR AI
    public class Human : GameObject
    {
```

```
#region Fields

public enum CharacterMotion
{
    Walking,
    Jumping,
    Orbing,
    Shooting,
    Grabbing,
    Falling,
    Idle
}

#endregion

#region Initialization
public Human(Game game)
    : base(game)
{
    Game.Services.AddService(typeof(Human), this);
}

public void CollisionResponse(object sender, EventArgs e)
{
}
public override void Initialize()
{
    base.Initialize();
}

public void LoadContent()
{
    base.LoadContent();
}
#endregion

#region Update

/// <summary>
/// Update movement
/// </summary>
/// <param name="gameTime"></param>
public override void Update(GameTime gameTime)
{
   base.Update(gameTime);
}

#endregion

#region Draw

//if we want animating stuff, it might get called here.
public override void Draw(GameTime gameTime)
{
    base.Draw(gameTime);
}
```

```
        #endregion
```

HUD

```
namespace GameOrDie.HUDisplay
{
    /// <summary>
    ///  Creates, updates, and draws a HUD on the user's viewport.
    /// </summary>
    public class HUD : DrawableGameComponent
    {
        #region Init and Update

        private List<TextDisplay> _textDisplays = new
List<TextDisplay>();
        public List<ImageDisplay> _imageDisplays = new
List<ImageDisplay>();

        /// <summary>
        /// Heads Up Display constructor.
        /// </summary>
        public HUD(Game game) : base(game)
        {
           CreateHUD();
           InitializeDialogControl();
        }

        protected override void LoadContent()
        {
            base.LoadContent();
        }

        public override void Update(GameTime gameTime)
        {
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
        }
        /// <summary>
        /// Update Hud events.
        /// </summary>
        public void Update()
        {
            UpdateDialogue();
        }
        #endregion

        #region HUDCreation
        /// <summary>
        /// Generate the HUD from XML file
        /// </summary>
```

```
        public void CreateHUD()
        {
            //  Insert creation code and then resize for correct
locations
            FindHudMultiplier();
        }

        private void FindHudMultiplier()
        {
            // Finds the resize ratio of the current dimensions/actual
            // viewport
        }
        #region Dialogue Controls
        /// <summary>
        /// Inits the dialogue control system
        /// </summary>
        public void InitializeDialogControl()
        {

        }
        /// <summary>
        /// Shows the dialogue
        /// </summary>
        public void ShowDialog()
        {
        }
        /// <summary>
        /// Removes the dialogue from the HUD
        /// </summary>
        public void RemoveDialog()
        {
        }
        /// <summary>
        /// Displays the dialogue on the HUD
        /// </summary>
        public void DisplayDialog()
        {
        }
        /// <summary>
        /// Removes dialogue once time limit expires
        /// </summary>
        public void UpdateDialogue()
        {
        }
        #endregion

        #region Add images/text to HUD and helper methods

        /// <summary>
        /// Add a Text display to the TextDisplay list.
        /// </summary>
        public int AddTextDisplay()
        {
        }

        /// <summary>
```

```
        /// Add an Image display to our ImageDisplay list.
        /// </summary>
      public void AddImageDisplay()
        {
        }
        /// <summary>
        /// Resizes the HUD based on viewport change
        /// </summary>
        /// <param name="HudScaleMultiplier">Old viewport/new</param>
        public void Resize(Vector2 HudScaleMultiplier)
        {
            /// For each display in HUD, multiply the position of the
            display /// by the HudScaleMultiplier
        }

        #endregion

        #region Draw functions
        /// <summary>
        /// Draw function for drawing our displays
        /// NOTE: SpriteBatch Begin and End are handled outside this
scope.
        /// </summary>
        public void Draw()
        {
            foreach (ImageDisplay element in _imageDisplays)
            {
              // Insert draw call
            }

            foreach (TextDisplay element in textDisplays)
            {
                  // Insert draw call
            }

        }
        #endregion
    }
}
```

ScreenObject

```
namespace GameOrDie.HUDisplay
{
    /// <summary>
    /// Base class for all types.
    /// </summary>
    public class ScreenObject
    {
      // Shared Properties.
    };
    #region Children of ScreenObject
    /// <summary>
    /// Indicates what type of Heads Up Display element this is.
```

```
    /// </summary>
    public enum DisplayType
    {
        /// <summary>
        /// A string of text.
        /// </summary>
        TEXT,
        /// <summary>
        /// An image.
        /// </summary>
        IMAGE,
        /// <summary>
        /// Default.
        /// </summary>
        BASE
    };

    /// <summary>
    /// Used to display Text.
    /// </summary>
    public class TextDisplay : ScreenObject
    {
      // Text display properties.
    }
    /// <summary>
    /// Used to display Images.
    /// </summary>
    public class ImageDisplay : ScreenObject
    {
      // Text display properties.

    }
    #endregion
}
```

XML Loading Schema

```
        // File loction, new XML reader
        string fileName += "\\_xml\\name_of_file";
        XmlDocument reader = new XmlDocument();
        reader.Load(fileName);

        XmlNodeList allNodes = reader.ChildNodes;
        // Go through all parent XMLNodes
        foreach (XmlNode eventNode in allNodes)
        {
          if (eventNode.Name == "Node_Name")
           {
             // If parent node, go through children
             XmlNodeList eventNodes = eventNode.ChildNodes;
             foreach (XmlNode node in eventNodes)
               {
                if (node.Name == "Class_or_Data_Type_Name")
                  {
```

```
                    // XML Loading of simple data types
                    this.[string] = node.Attributes["texture"].Value;
                    this.[bool] =
bool.Parse(node.Attributes["bool"].Value);
                    this.[int] =
int.Parse(node.Attributes["int"].Value);

                    // XML Loading of Vector2 data type
                    this.[Vector2] = new
                      Vector2(int.Parse(node.Attributes["X"].Value),
                      int.Parse(node.Attributes["Y"].Value));

                    // XML Loading of Vector2 data type
                    this.[Vector4]= new Vector4 (
                       byte.Parse(node.Attributes["R"].Value),
                       byte.Parse(node.Attributes["G"].Value),
                       byte.Parse(node.Attributes["B"].Value),
                       byte.Parse(node.Attributes["A"].Value)
                       );
                    }
                }
            }
        }
```

Design History

Version	Release Date	Comments
1.0	1.X.2008	Initial release

[i] © Intel Corporation
[ii] © Advanced Micro Devices, Inc.
[iii] © Microsoft Corporation
[iv] © Microsoft Corporation